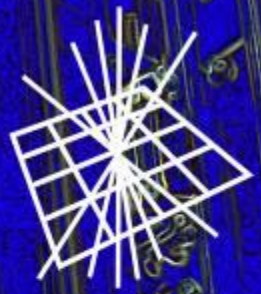


GridPP

UK Computing for Particle Physics

Multi-core jobs at the RAL Tier-1

Andrew Lahiff, Alastair Dewhurst, John Kelly

February 25th 2014

Introduction

- RAL supports many VOs
  - ALICE, ATLAS, CMS, LHCb
  - ~12 non-LHC VOs
- Size of batch system
  - Currently over 9300 job slots
- Used Torque/Maui for many years
  - Reliability/stability degraded as number of cores/WNs increased
- Migrated to HTCondor last year
  - Started running production ATLAS/CMS jobs in June, parallel to production Torque/Maui
  - In November fully moved to HTCondor
- Multi-core usage
  - ATLAS almost continuously since November
  - CMS have run a few test jobs
  - No interest so far from other VOs

Job submission

Submission of multi-core jobs to RAL

- ATLAS and CMS only using ARC CEs (no CREAM!)
- Configuration
  - Setup with only a single queue
  - Any VO can run multi-core jobs if they want to
    - Just have to request > 1 CPU. E.g. add to XRSL/GlobusURL:

      `(count=8)`
    - Any number of CPUs can be requested (2, 4, 8, 31, …)
      - But a job requesting 402 CPUs, for example, is unlikely to run ☺
  - Memory requirements
    - Jobs also request how much memory they need, e.g.

      `(memory=2000)`
    - For multi-core jobs this is the memory required (in MB) per core
      - Total memory required passed from CE to HTCondor

Worker node configuration

- Multi-core worker nodes in HTCondor – 3 possibilities
  1. Divide resources evenly (default)
     - 1 core per slot
     - Memory divided equally between slots
     - Fixed number of slots
  2. Define slot types
     - Specify (static) resources available to each slot, don't need to be divided equally
     - Fixed number of slots
  3. Partitionable slots

     (see next slide)
- Clearly
  - Choices 1 & 2 not suitable for environments where jobs have different memory & core requirements

Worker node configuration

- We're using partitionable slots
  - All WNs configured to have partitionable slots
  - Each WN has a *single partitionable slot* with a fixed set of resources (cores, memory, disk, swap, …)
  - These resources can then be divided up as necessary for jobs
    - Dynamic slots created from the partitionable slot
    - When dynamic slots exit, merge back into the partitionable slot
    - When any single resource is used up (e.g. memory), no more dynamic slots can be created
  - Enables us to run jobs with different memory requirements, as well as jobs requiring different numbers of cores
  - Our configuration:

```
NUM_SLOTS = 1
SLOT_TYPE_1                = cpus=100%,mem=100%,auto
NUM_SLOTS_TYPE_1           = 1
SLOT_TYPE_1_PARTITIONABLE = TRUE
```

Fairshares

- Initially just had a single accounting group for all ATLAS production jobs
  - Single & multi-core jobs in the same accounting group
  - Problem: fairshare could be maintained by just running single core jobs
- Added accounting groups for ATLAS & CMS multi-core jobs
- Force accounting groups to be specified for jobs using SUBMIT_EXPRS on CEs
  - Easy to include groups for multi-core jobs, e.g.:

```
AccountingGroup =
…
    ifThenElse(regexp("patl",Owner) && RequestCpus > 1,  "group_ATLAS.prodatls_multicore", \
    ifThenElse(regexp("patl",Owner),                     "group_ATLAS.prodatls", \
…
SUBMIT_EXPRS = $(SUBMIT_EXPRS) AccountingGroup
```

Defrag daemon

- If lots of single core jobs are idle & running, how does a multi-core job start?
  - By default it probably won't
- condor_defrag daemon
  - Finds worker nodes to drain, drains them & cancels draining when necessary
    - drain = no new jobs can start but existing jobs continue to run
  - Many configurable parameters, including:
    - How often to run
    - Maximum number of machines running multi-core jobs
    - Maximum concurrent draining machines
    - Expression that specifies which machines to drain
    - Expression that specifies when to cancel draining
    - Expression that specifies which machines are already running multi-core jobs
    - Expression that specifies which machines are more desirable to drain

Defrag daemon

- Improvements to default configuration (1)
  - condor_defrag originally designed for enabling whole node jobs to run
    - We currently want to run 8-core jobs, not whole node jobs
  - Definition of “whole” machines
    - Only drain up to 8-cores, not entire machines
    - Changed the default:

      ```
      DEFRAG_WHOLE_MACHINE_EXPR = Cpus == TotalCpus && Offline=!=True
      ```

      to

      ```
      DEFRAG_WHOLE_MACHINE_EXPR = ((Cpus == TotalCpus) || (Cpus >= 8)) && Offline=!=True
      ```

      Here Cpus = free CPUs

Defrag daemon

- Improvements to default configuration (2)
  - Which machines are more desirable to drain
    - We assume that:
      - On average all jobs have the same wall time
      - Wall time used by jobs is not related to the wall time they request
    - Weighting factor

      Number of cores available for draining / Number of cores that need draining
    - Changed the default:

      `DEFRAG_RANK = -ExpectedMachineGracefulDrainingBadput`

      The job runtime in cpu-seconds that would be lost if graceful draining were initiated at the time the machine's ad was published. Assumes jobs will run to their walltime limit.

      to:

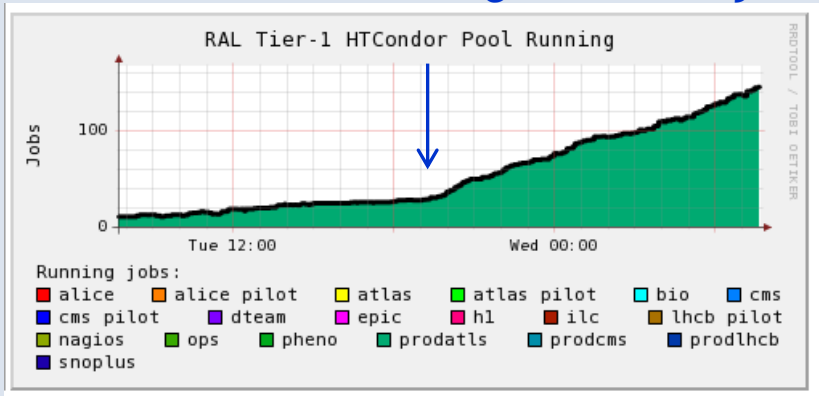
      Used CPUs / (8 - Free CPUs)

      `DEFRAG_RANK = ifThenElse(Cpus >= 8, -10, (TotalCpus - Cpus)/(8.0 - Cpus))`
    - Why make this change?
      - Previously only older 8-core machines were selected for draining
      - Now machines with the most numbers of cores are selected for draining
        - » 8-cores become available more quickly

Defrag daemon

- Effect of change to DEFRAG_RANK

Running multi-core jobs



- Effect of switching off condor_defrag
  - Will number of running multi-core jobs quickly reduce to zero?
  - Number of running multi-core jobs decreased slowly

Cancelled all draining worker nodes

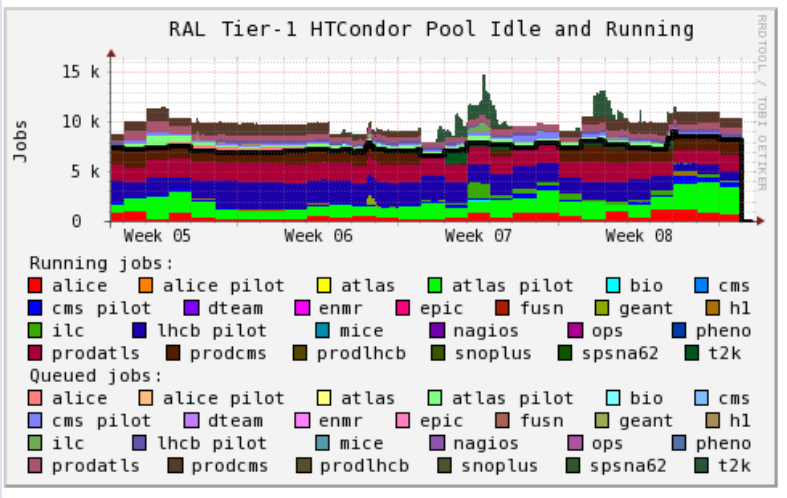
Conclusion:

Need to drain slots continuously to maintain number of running multi-core jobs

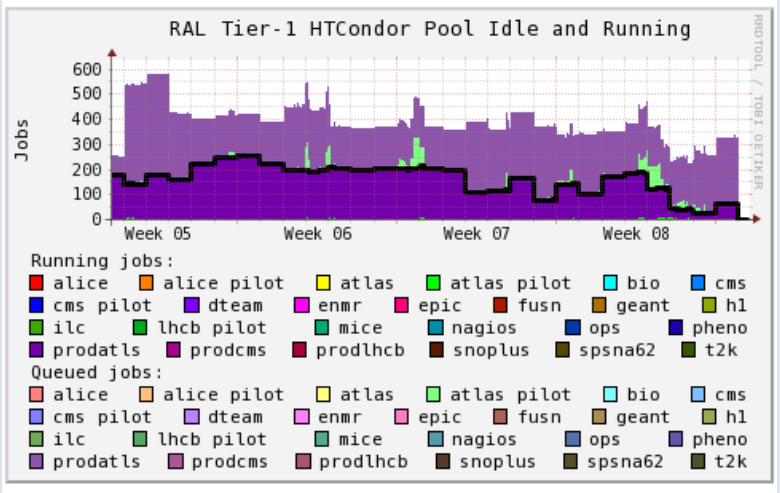
Recent jobs

Past month

Jobs idle/running



Multi-core jobs idle/running



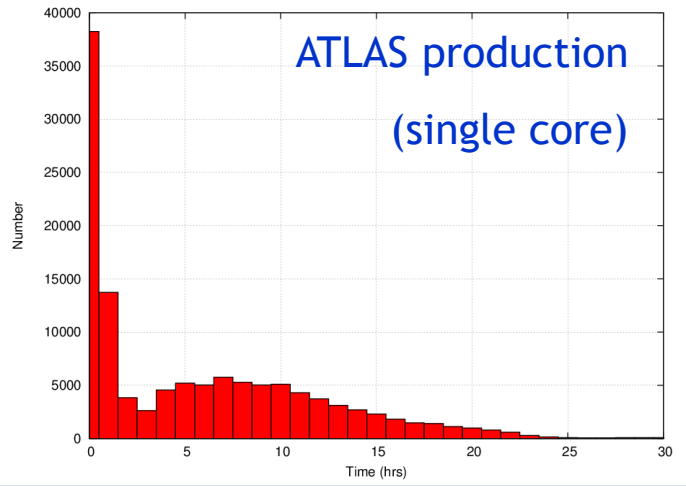
Limit of number of running ATLAS multi-core jobs has been their fairshare

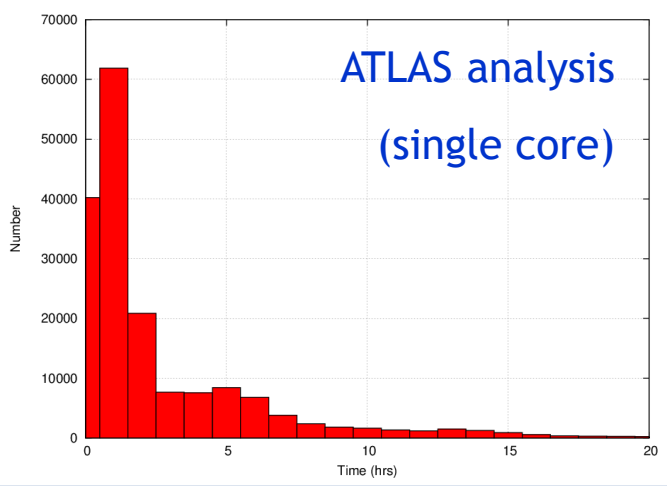
Originally saw a sawtooth pattern of running multi-core jobs

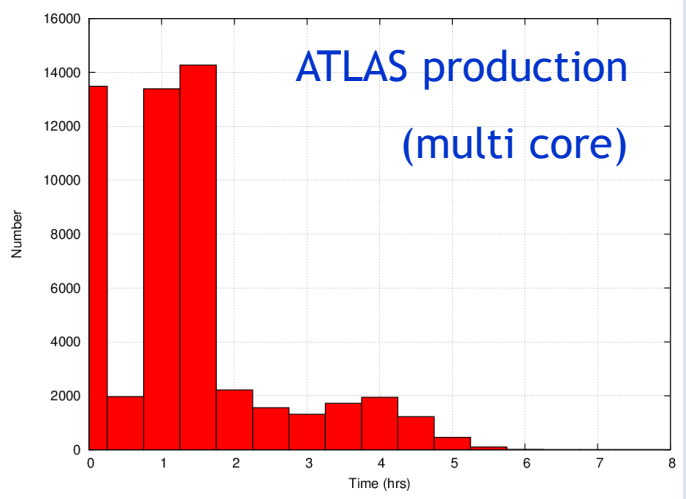
- ATLAS then made a change to keep a smoother supply of multi-core jobs

Recent jobs

Wall times since 1st Feb





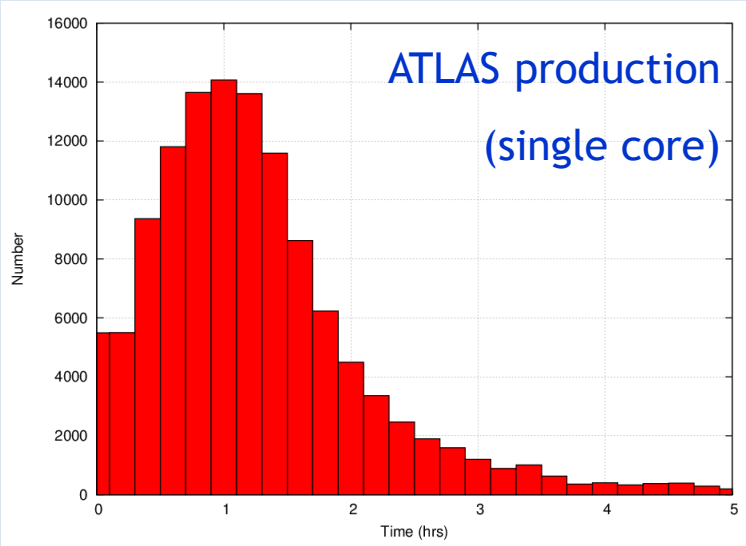


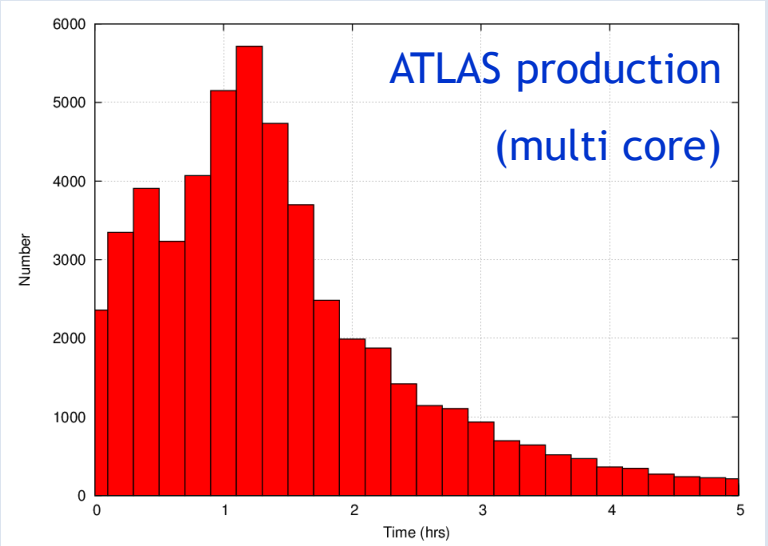
ATLAS ‘timefloor’ parameter

- Analysis pilots running < 1 hour will pick up another job
- To help with draining, might change this or add a short analysis queue
- Might add timefloor parameter to multi-core queue so that the slots are kept longer

Recent jobs

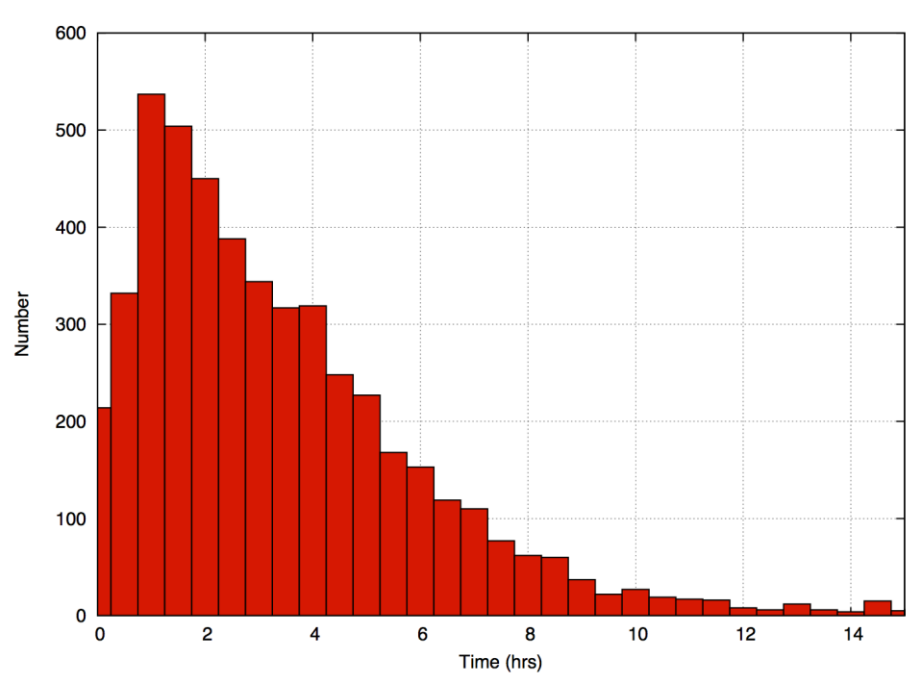
Wait times since 1st Feb





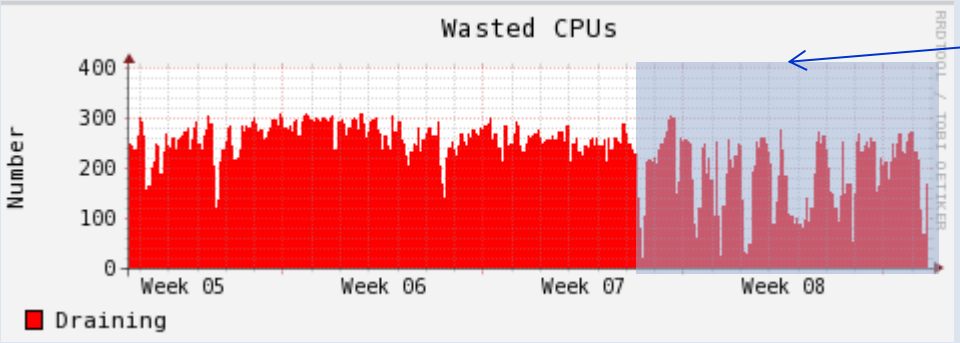
Recent jobs

Time to drain 8 slots



Wasted resources

- Defrag daemon is currently quite simple
  - Not aware of demand (e.g. idle multi-core jobs)
  - Drains a constant number of worker nodes all the time
- Added monitoring of wasted CPUs due to draining
  - Past month



Attempting to reduce wasted resources (next slide)

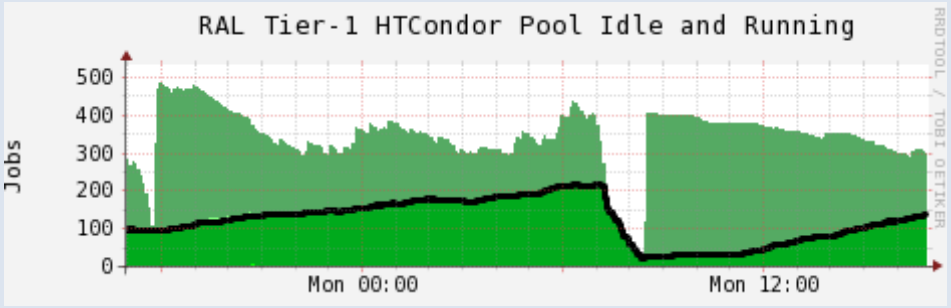
  - We an clearly see the wastage – it's not hidden within a multi-core pilot running a mixture of single & multi-core jobs
- Important question: can we reduce the number of wasted CPUs?

Wasted resources

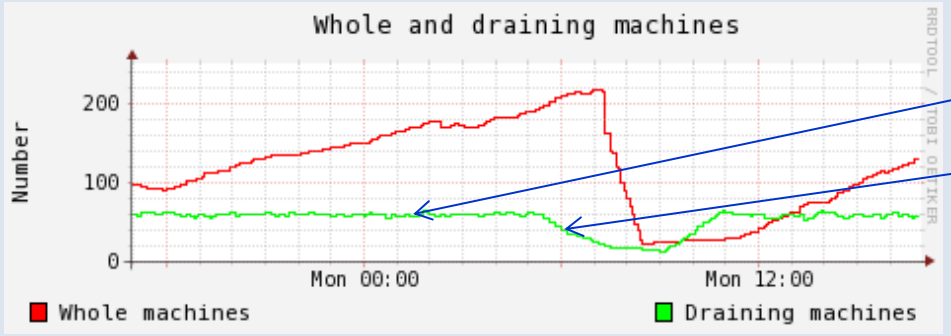
- Method to reduce wasted resources
  - Cron which sets parameters of defrag daemon based on running & idle multi-core jobs
    - In particular number of concurrent draining WNs
    - Runs every 30 mins
  - 1st attempt very simple, considers 3 situations
    - Many idle multi-core jobs, few running multi-core jobs
      - Need aggressive draining
    - Many idle multi-core jobs, many running multi-core jobs
      - Need less aggressive draining – just enough to maintain number of currently running multi-core jobs
    - Otherwise
      - Very low amount of draining required

Wasted resources

- Example from 24th Feb

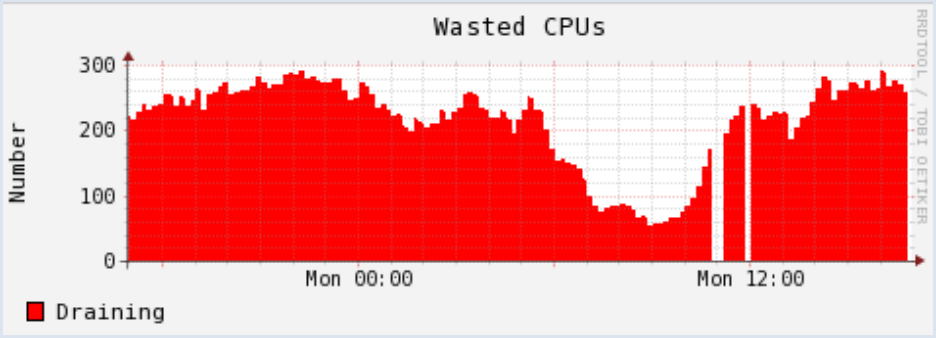


Multi-core jobs only shown



Aggressive draining

Less aggressive draining



Information system / accounting

- BDII hasn't been a priority
  - ATLAS and CMS don't even use it!
  - Currently:

```
$ ldapsearch -x -h arc-ce01.gridpp.rl.ac.uk -p 2135 -LLL -b o=grid | grep "GlueHostArchitectureSMPSize:"
GlueHostArchitectureSMPSize: 1
```

  - May fix this if someone needs it
- Accounting
  - ARC CEs send accounting records directly to APEL brokers
  - With latest ARC rpms multi-core jobs are handled correctly

Future plans

- Reducing wasted resources
  - Improvements to script which adjusts defrag daemon parameters
  - Make multi-core slots “sticky”
    - It takes time & resources to provide multi-core slots
    - After a multi-core job runs, prefer to run another multi-core job rather than multiple single core jobs
  - Try to run “short” jobs on slots which are draining?
- Multiple multi-core jobs per WN
  - By default, condor_defrag assumes “whole node” rather than “multi-core” jobs
  - Currently will have at most 1 multi-core job per WN
    (unless there aren’t many single core jobs to run)
  - Modify condor_defrag configuration as necessary to allow this